Pedestrian detection at 100 frames per second

R. Benenson, M. Mathias, R. Timofte and L. Van Gool

KATHOLIEKE UNIVERSITEIT LEUVEN

VISICS

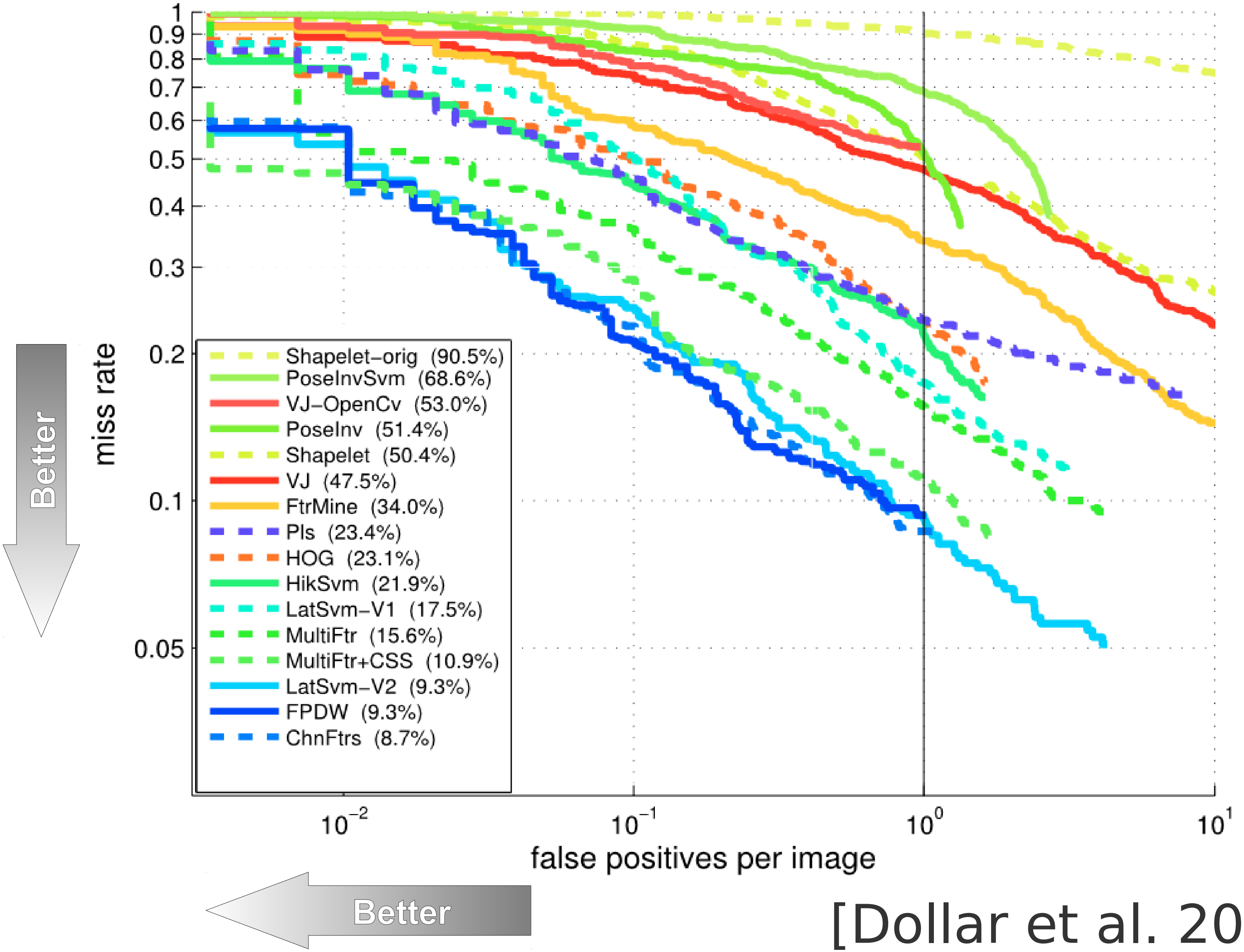
Why 100 fps ?

- Detection is one module amongst many
- Less computational power
- *Latency matters*

How can we make things faster ?

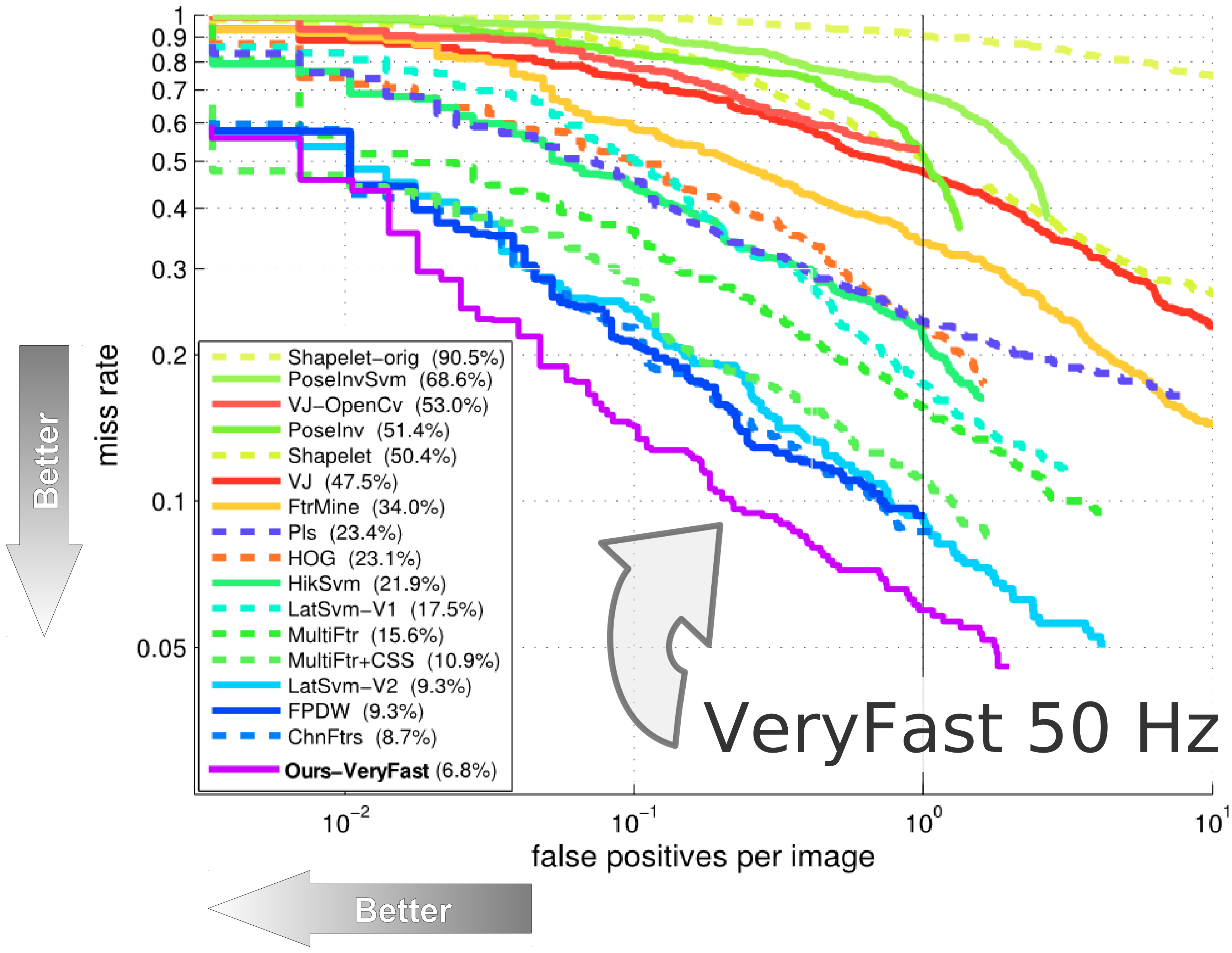
How can we make
things *algorithmically* faster ?

INRIA dataset



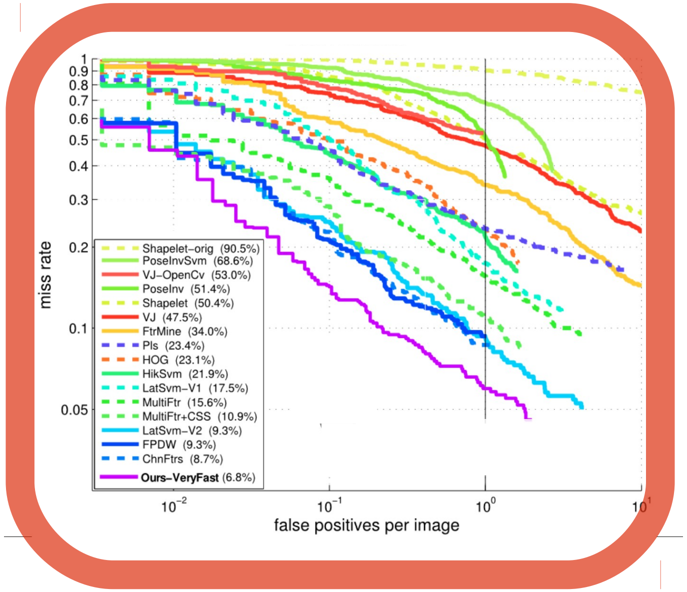
[Dollar et al. 2011]

INRIA dataset

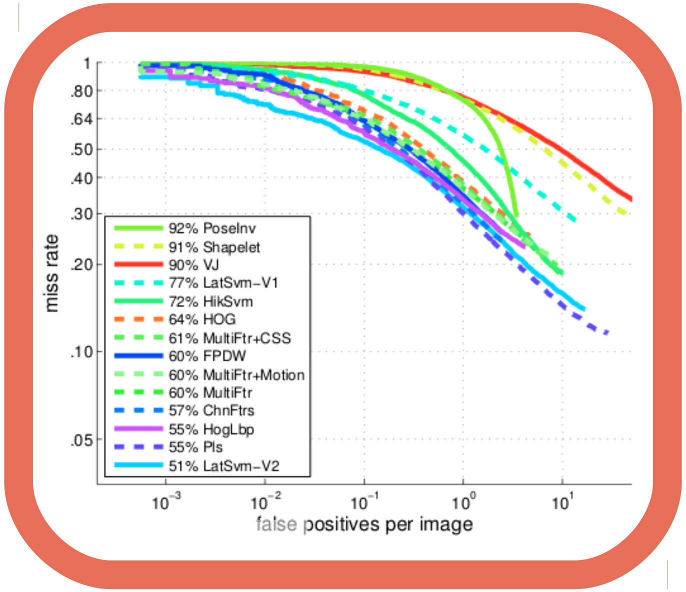


VeryFast 50 Hz

INRIA dataset

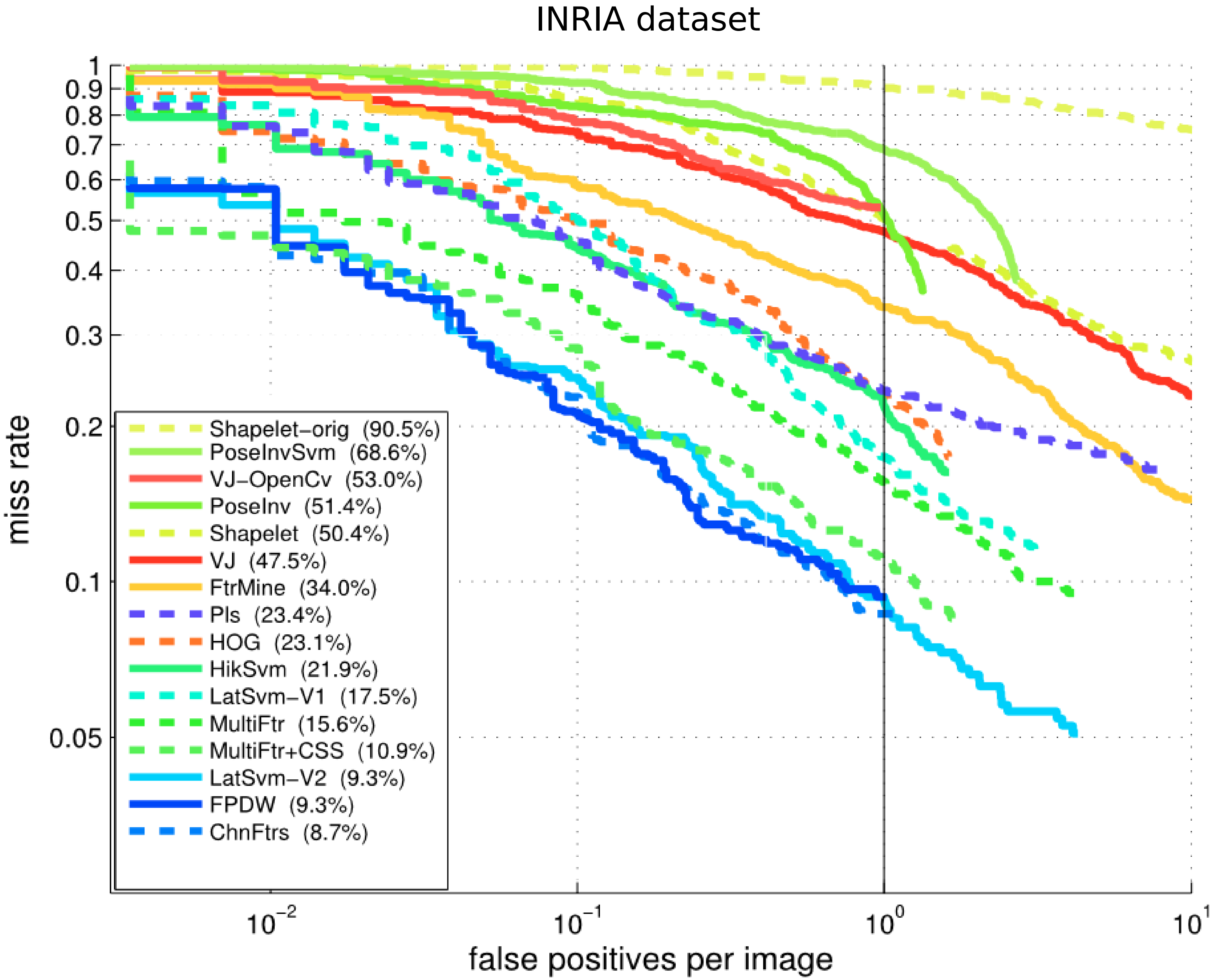


ETH dataset

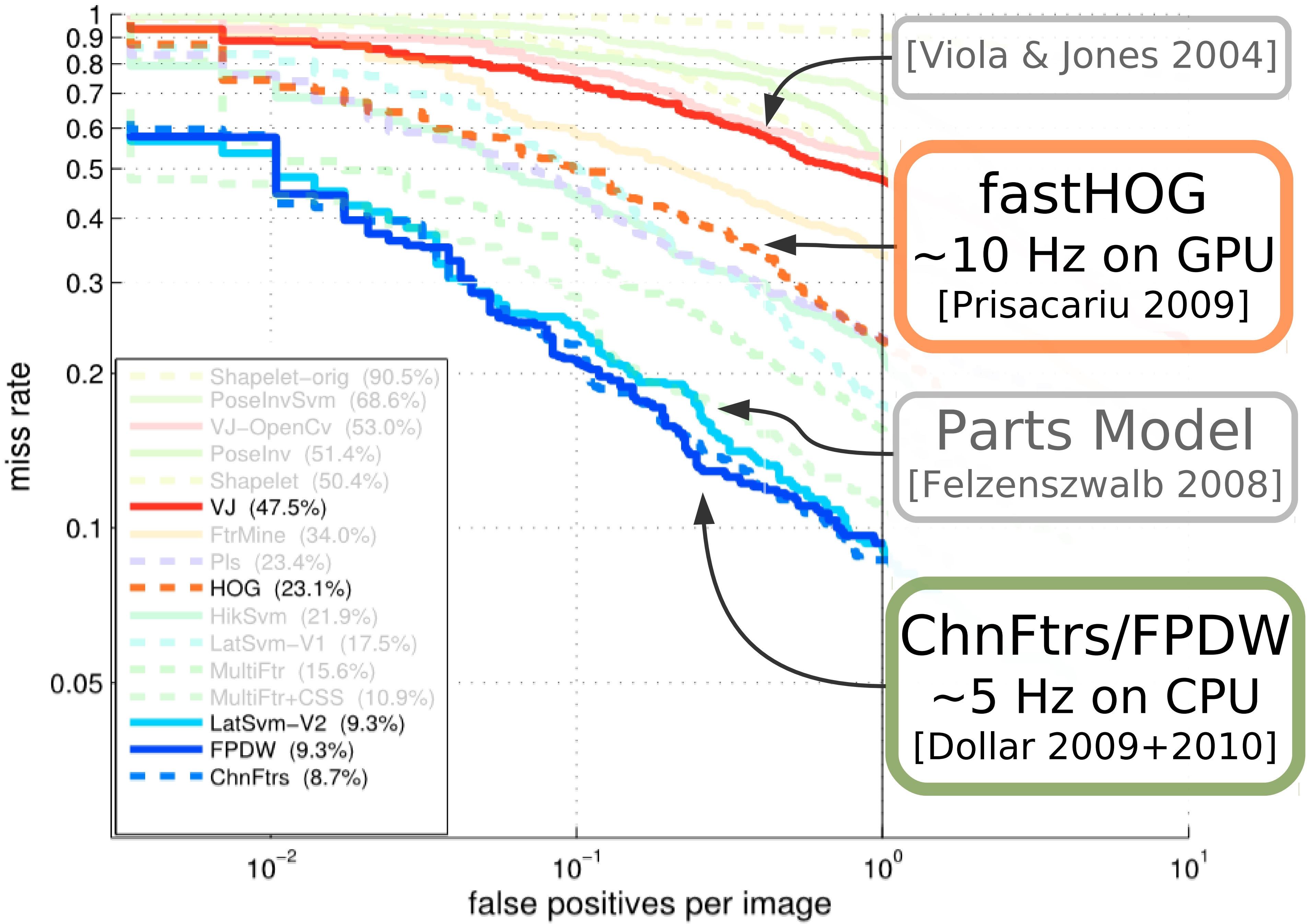


Monocular
50 Hz

Stereo
135 Hz


INRIA dataset
miss rate
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.05
false positives per image
10^{-2}
10^{-1}
10^{0}
10^{1}
Shapelet−orig (90.5%)
PoseInvSvm (68.6%)
VJ−OpenCv (53.0%)
PoseInv (51.4%)
Shapelet (50.4%)
VJ (47.5%)
FtrMine (34.0%)
Pls (23.4%)
HOG (23.1%)
HikSvm (21.9%)
LatSvm−V1 (17.5%)
MultiFtr (15.6%)
MultiFtr+CSS (10.9%)
LatSvm−V2 (9.3%)
FPDW (9.3%)
ChnFtrs (8.7%)

INRIA dataset

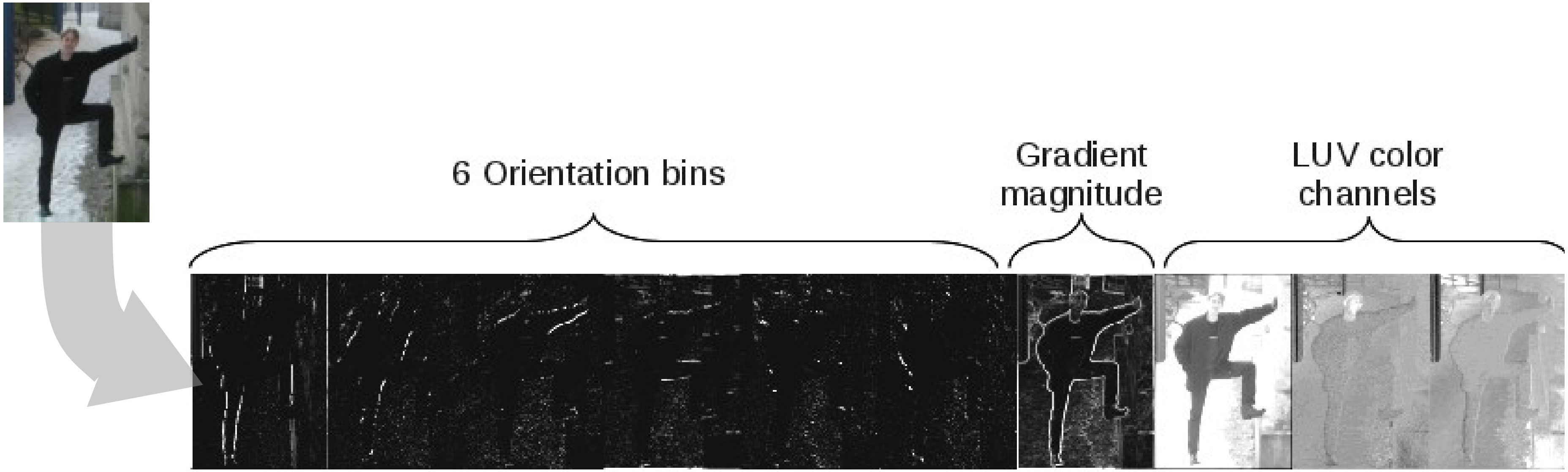


[Viola & Jones 2004]

fastHOG
~10 Hz on GPU
[Prisacariu 2009]

Parts Model
[Felzenszwalb 2008]

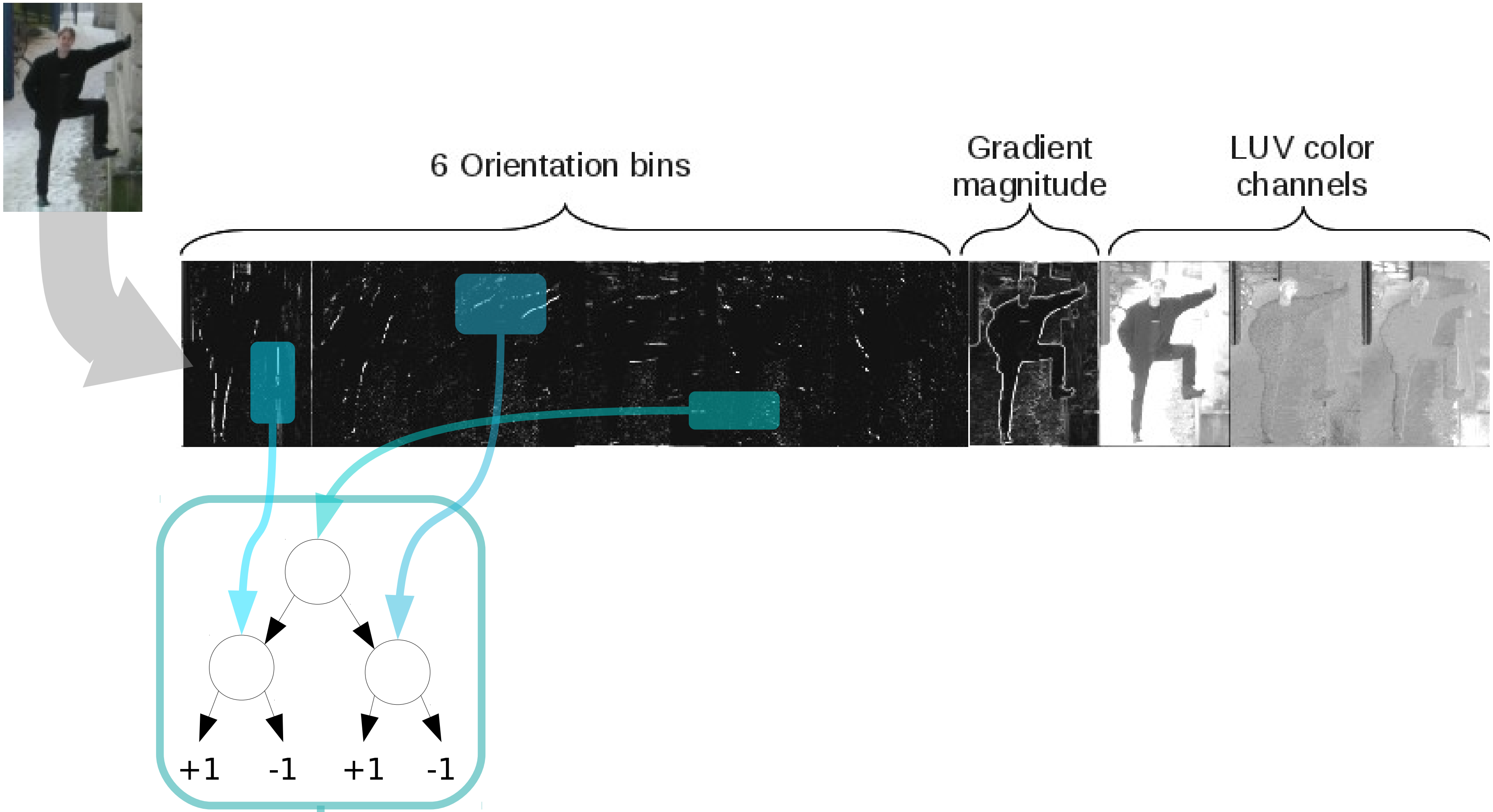
ChnFtrs/FPDW
~5 Hz on CPU
[Dollar 2009+2010]



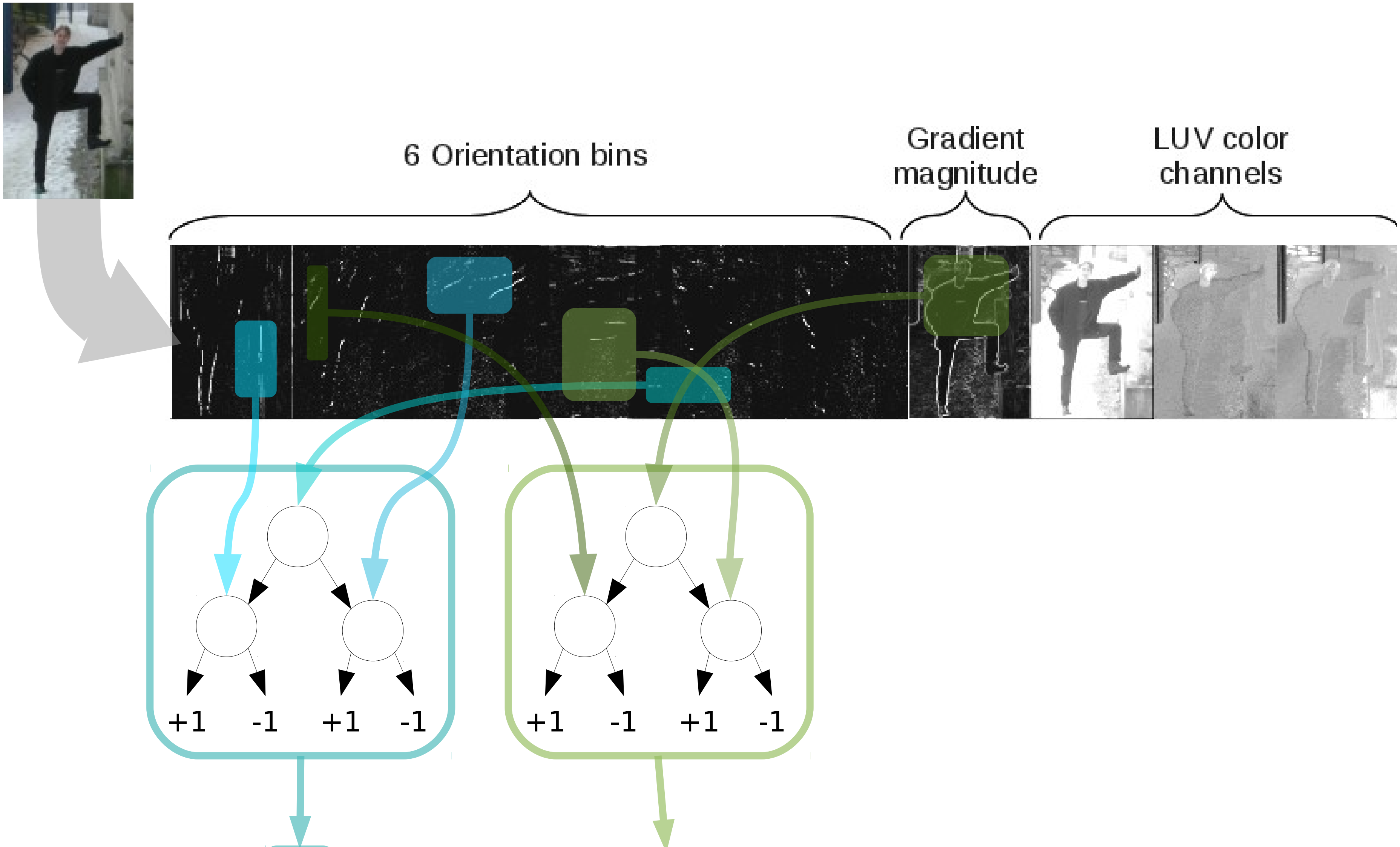
6 Orientation bins

Gradient magnitude

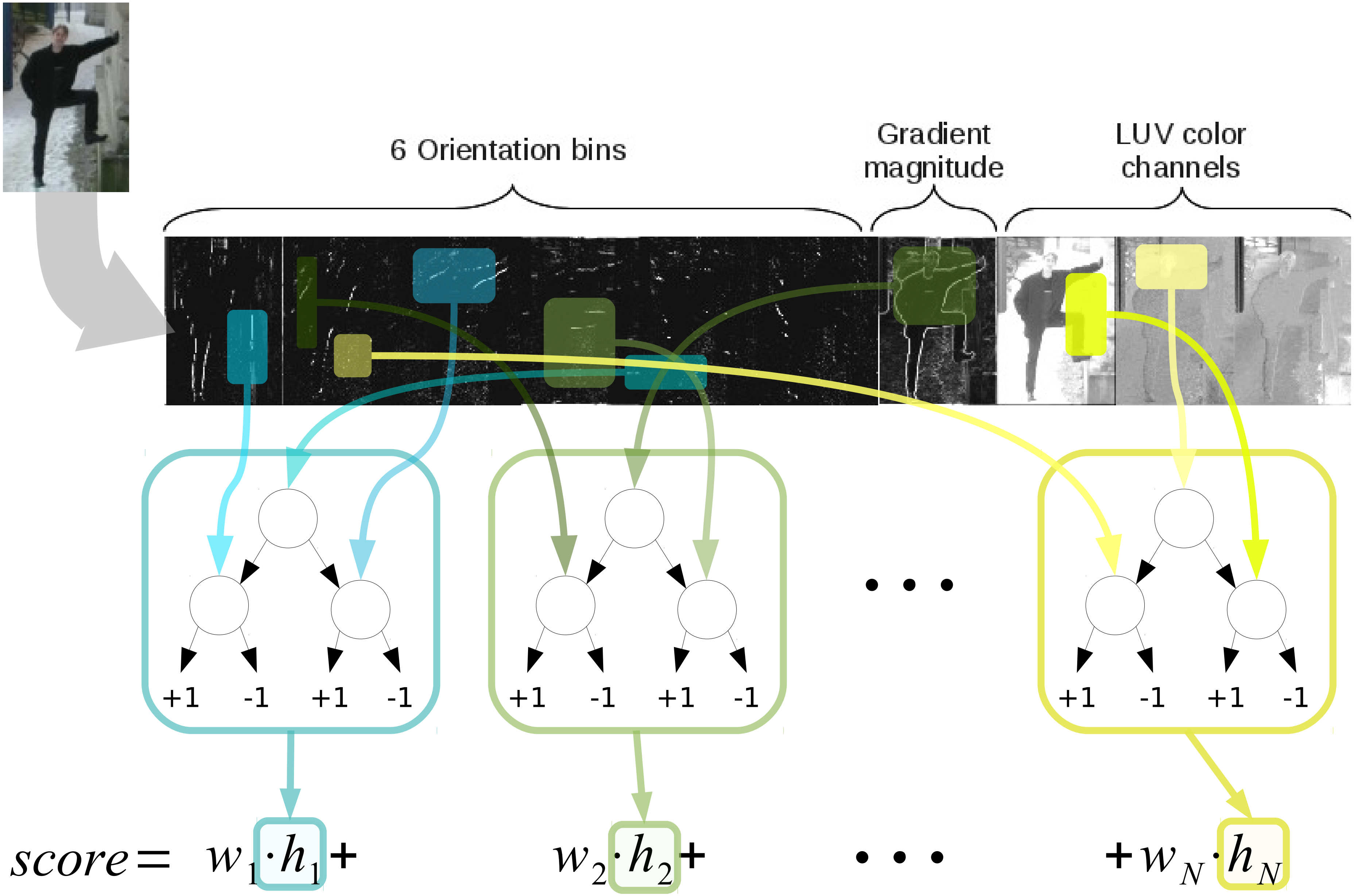
LUV color channels



$$score = w_1 \cdot \boxed{h_1} +$$



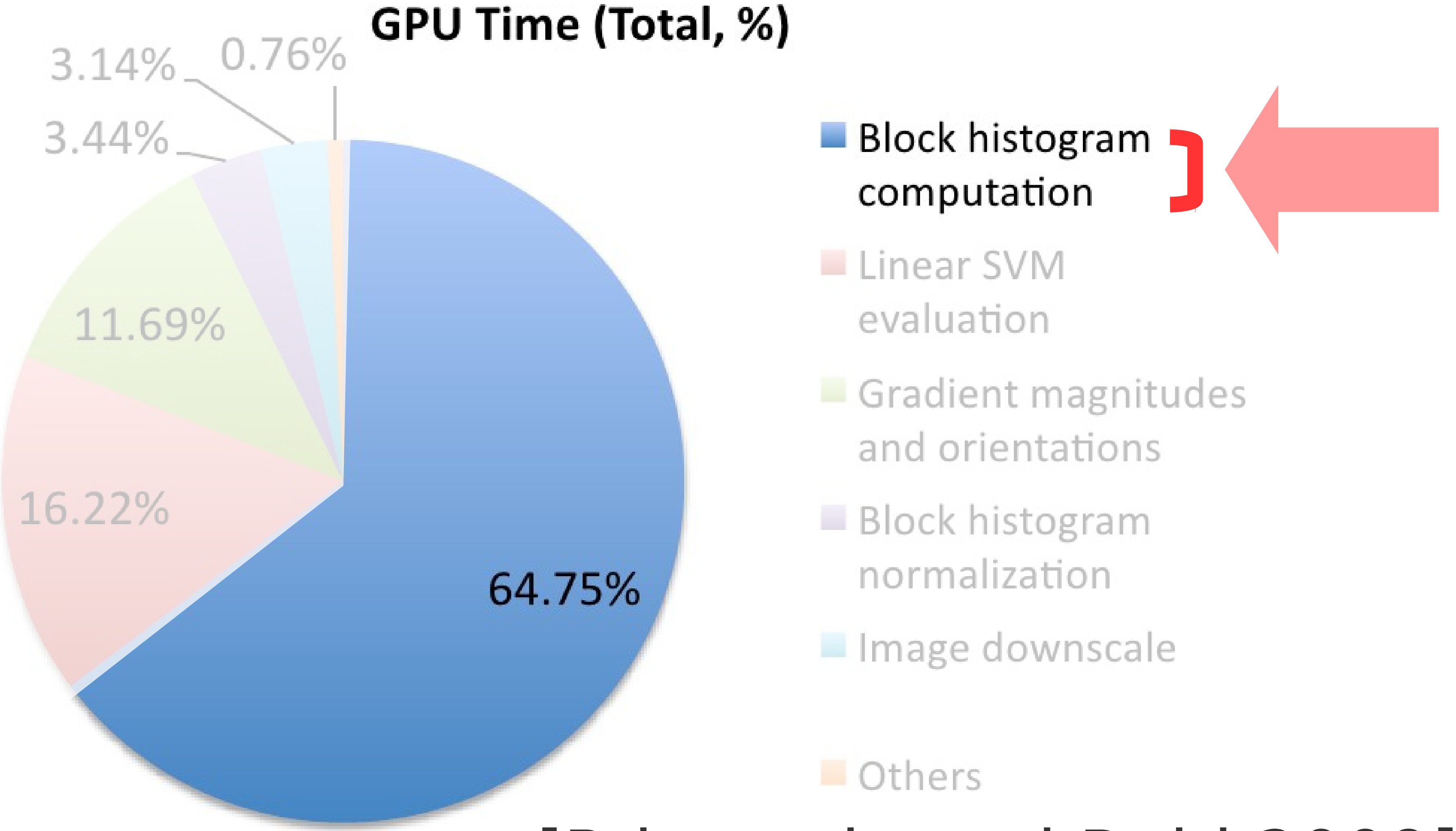
$$score = w_1 \cdot \boxed{h_1} + \quad w_2 \cdot \boxed{h_2} +$$



$$score = w_1 \cdot \boxed{h_1} + \; w_2 \cdot \boxed{h_2} + \; \cdots \; + w_N \cdot \boxed{h_N}$$

[ChnFtrs, Dollar et al. 2009] (~4 Hz on GPU)

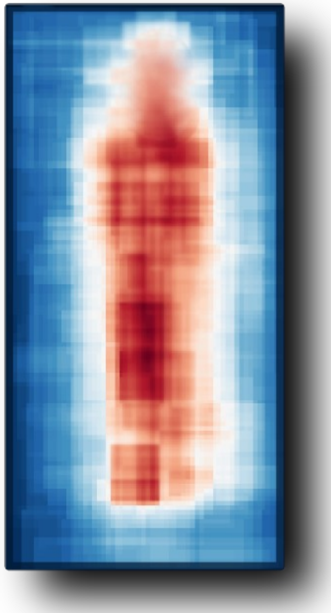
What slows down fastHOG ?

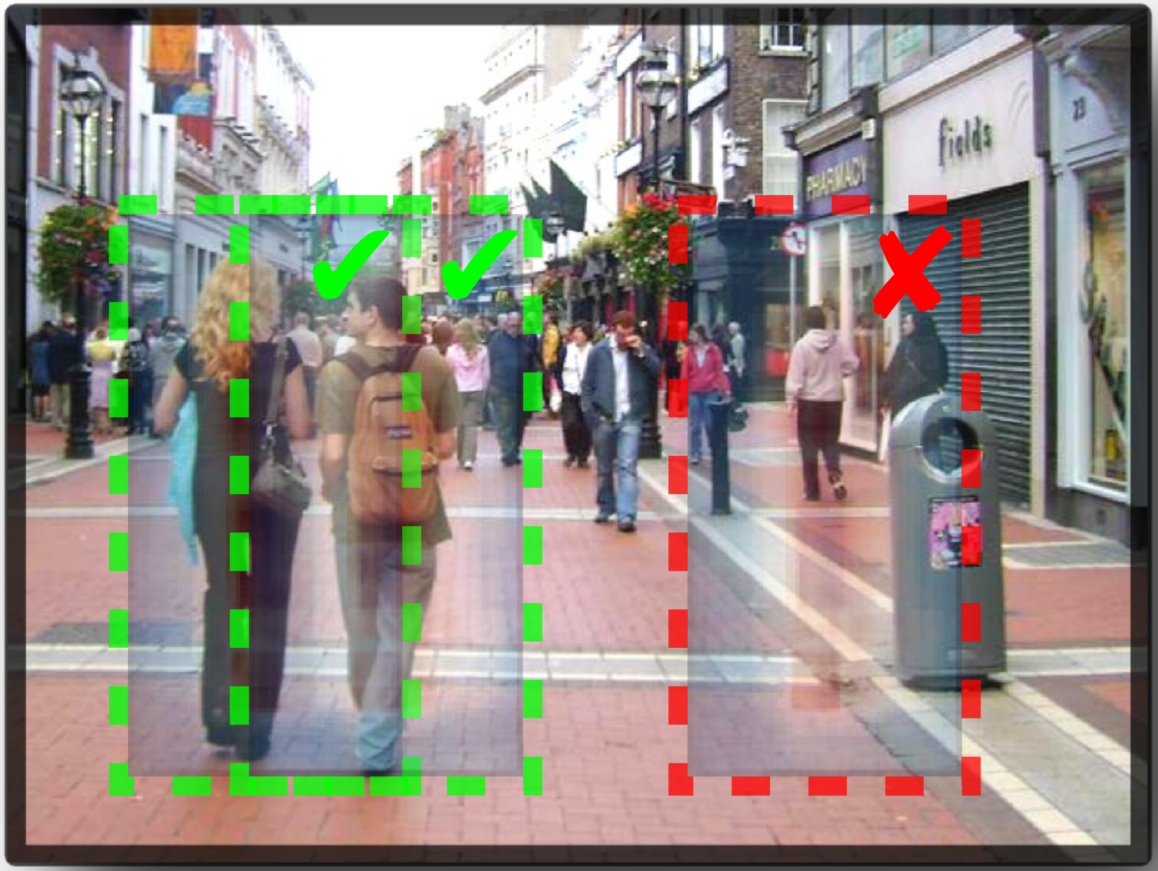


[Prisacariu and Reid 2009]

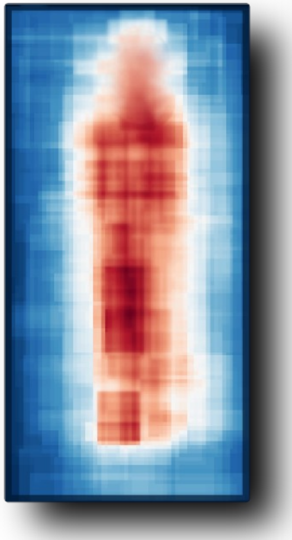
How to make *features computation* faster ?

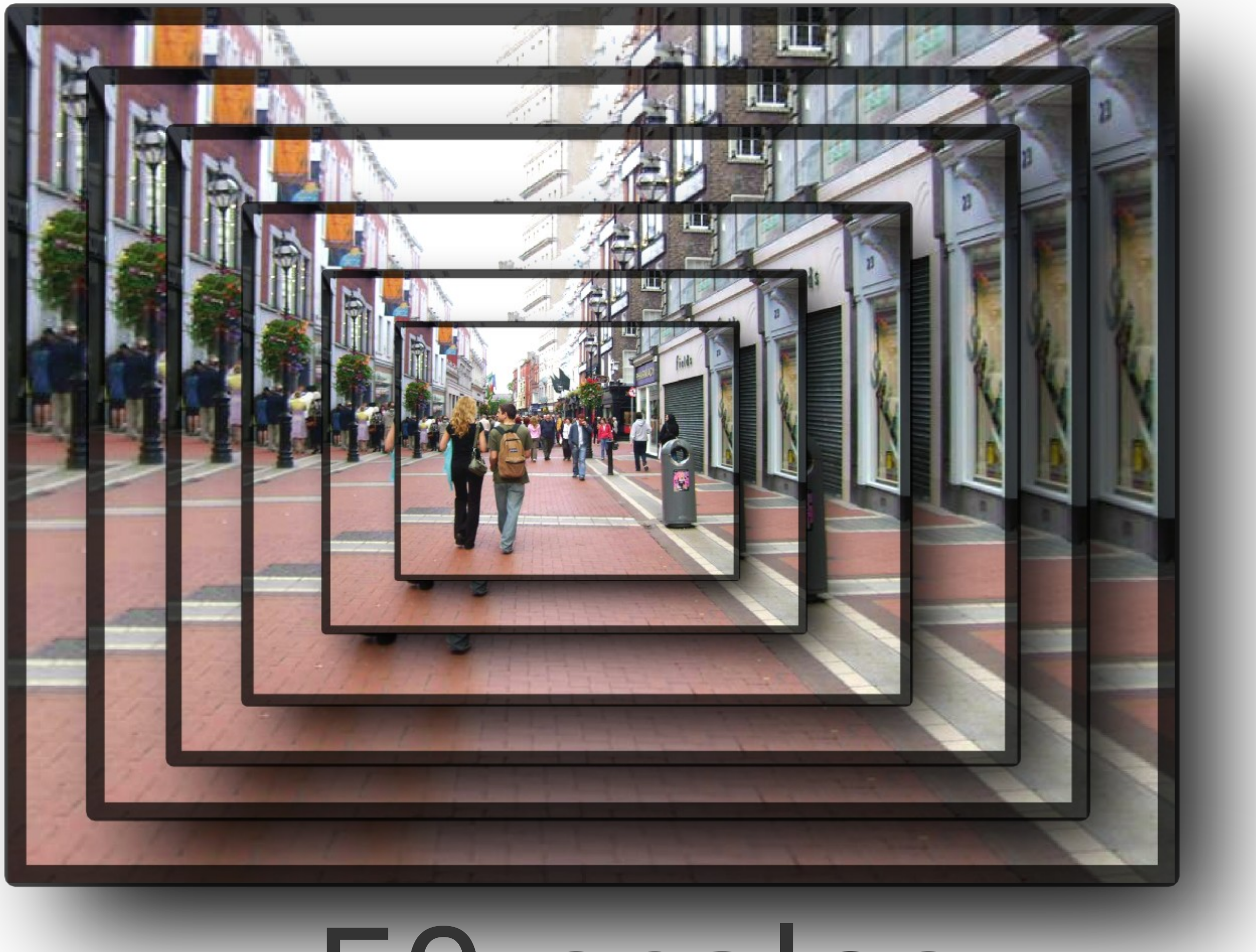
One template cannot detect at multiple scales





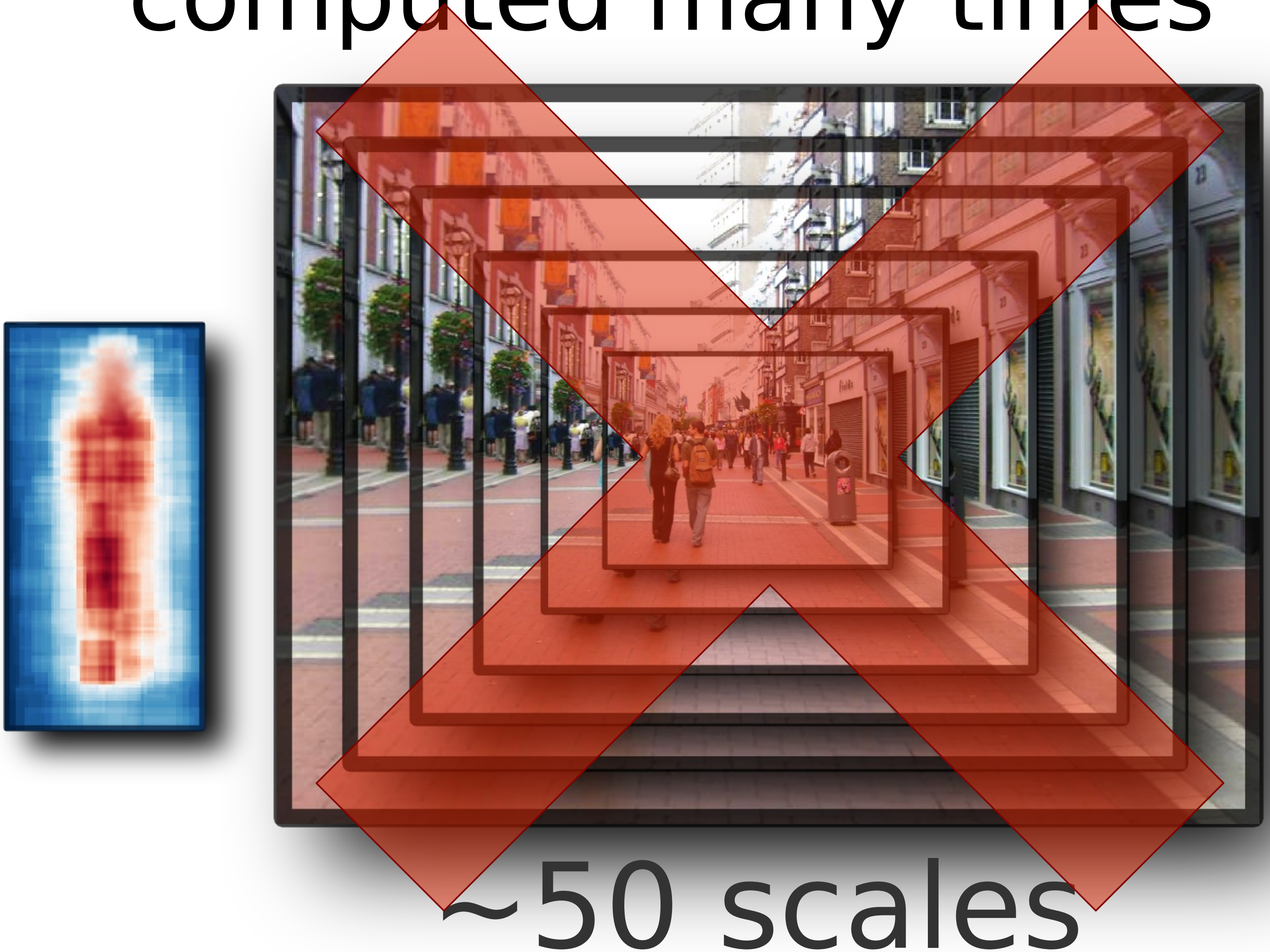
Traditionally, features are computed many times



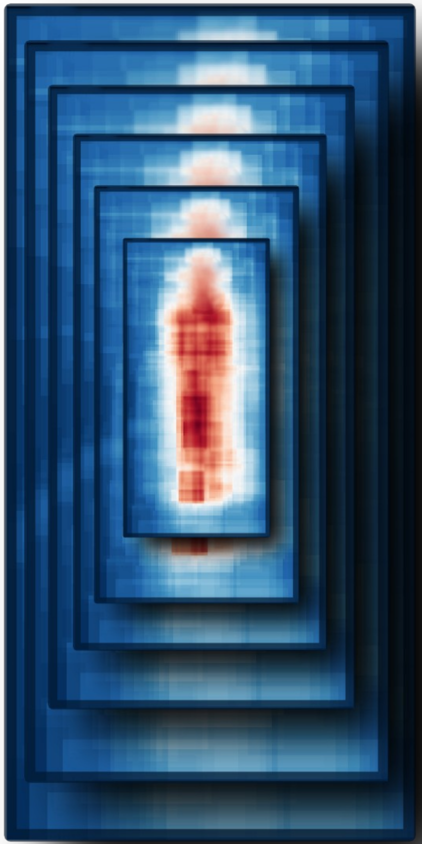


~50 scales

Traditionally, features are computed many times



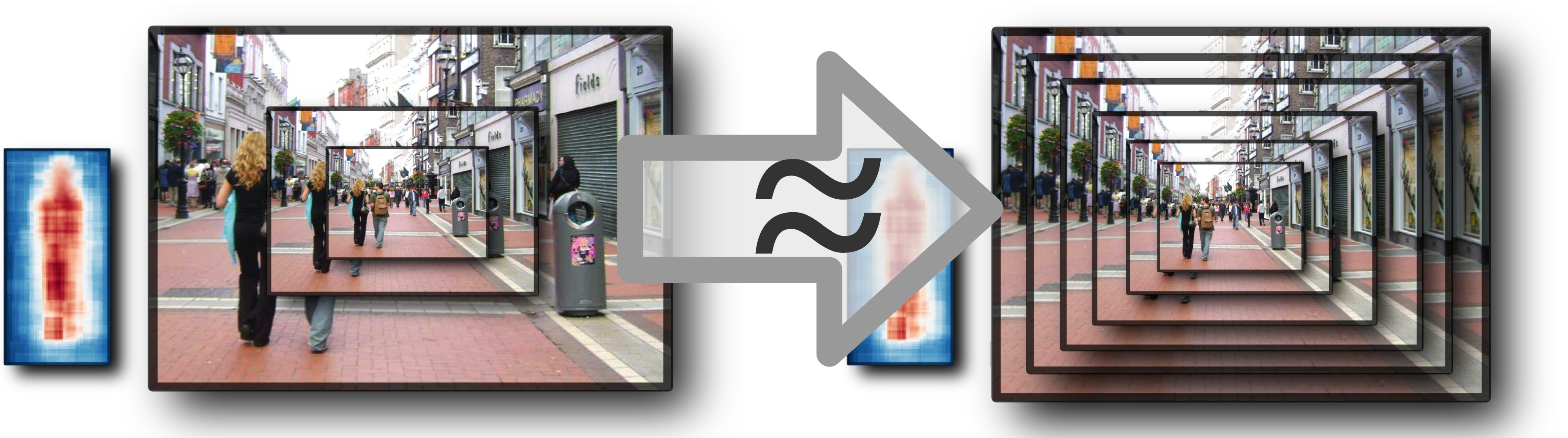
Training one model per scale is too expensive



~50 scales

Can we avoid resizing
the input image many times ?

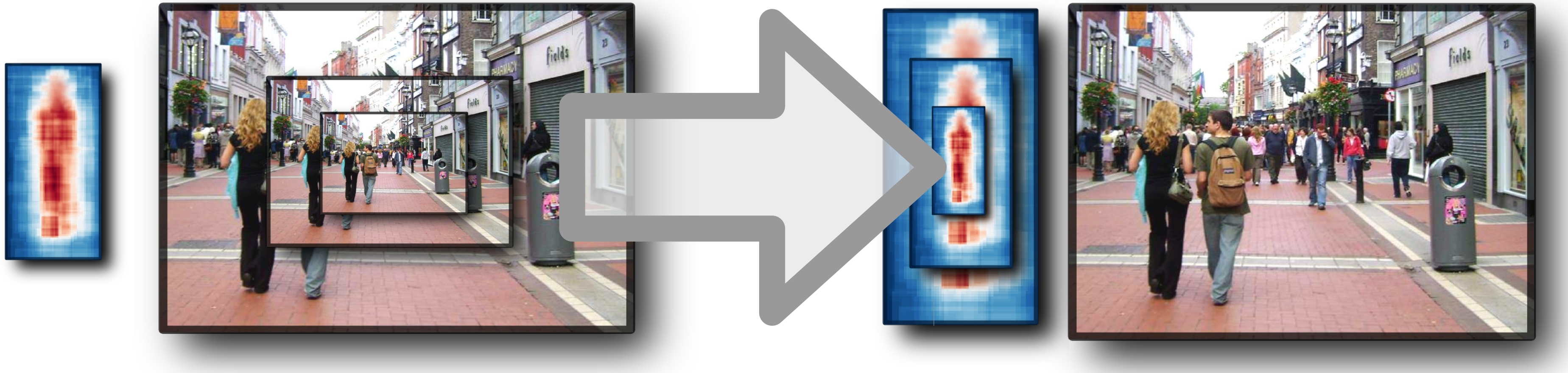
Features can be approximated across scales



~5 scales ~50 scales

[Dollar et al. 2010]

We transfer test time computation to training time

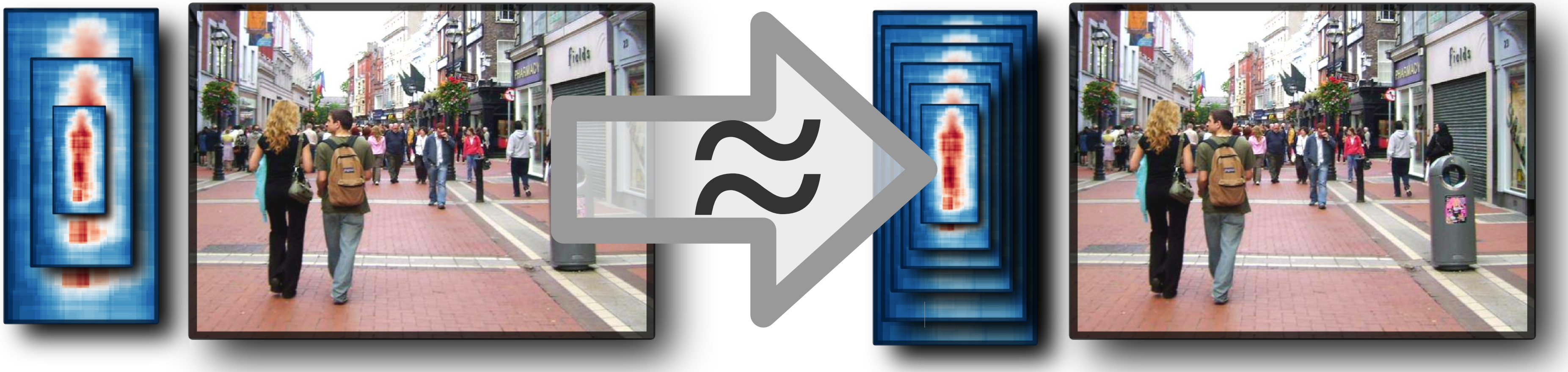


1 model,
5 image scales

5 models,
1 image scale

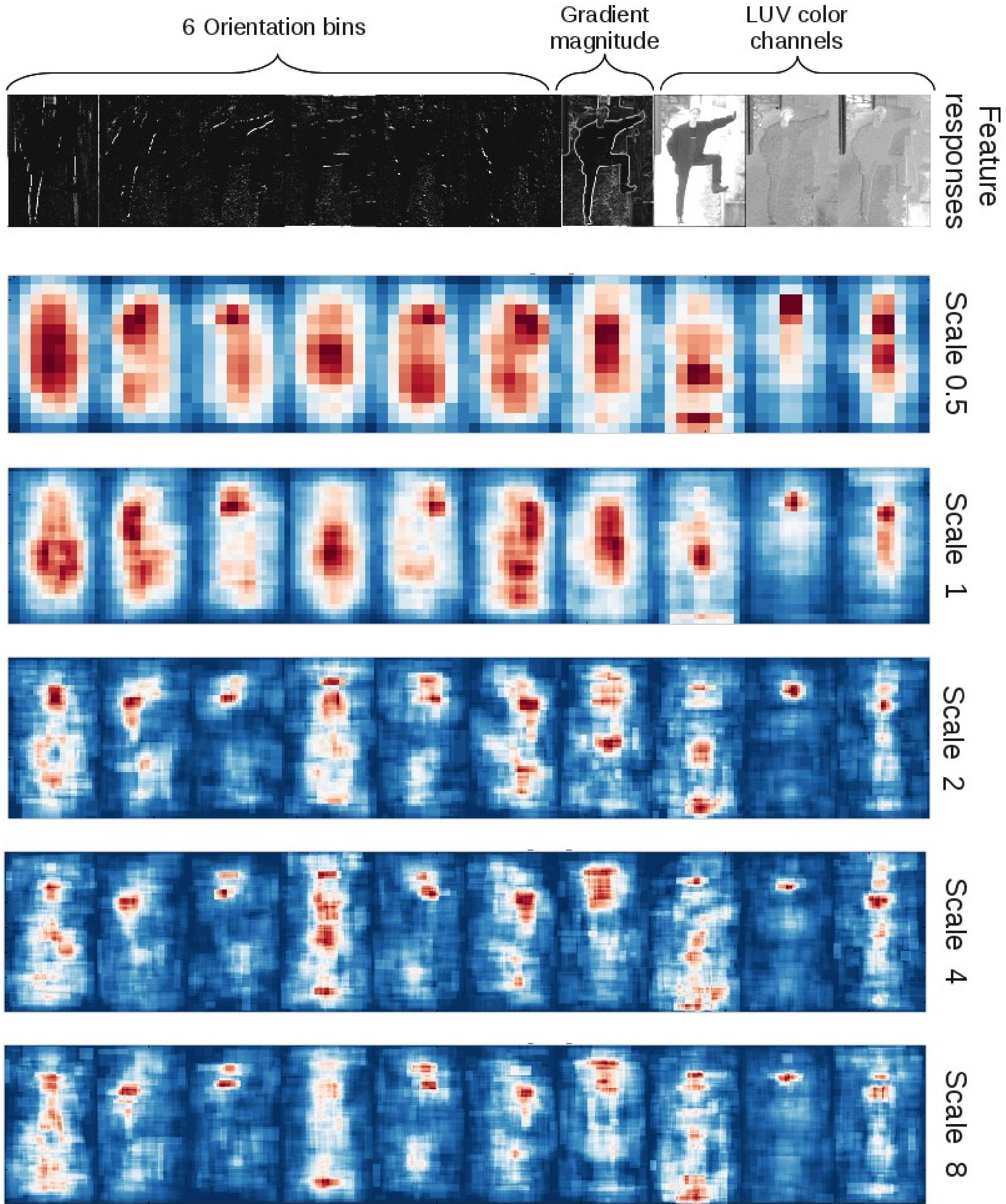
(3x reduction in features computation)

At runtime, we use as many models as scales

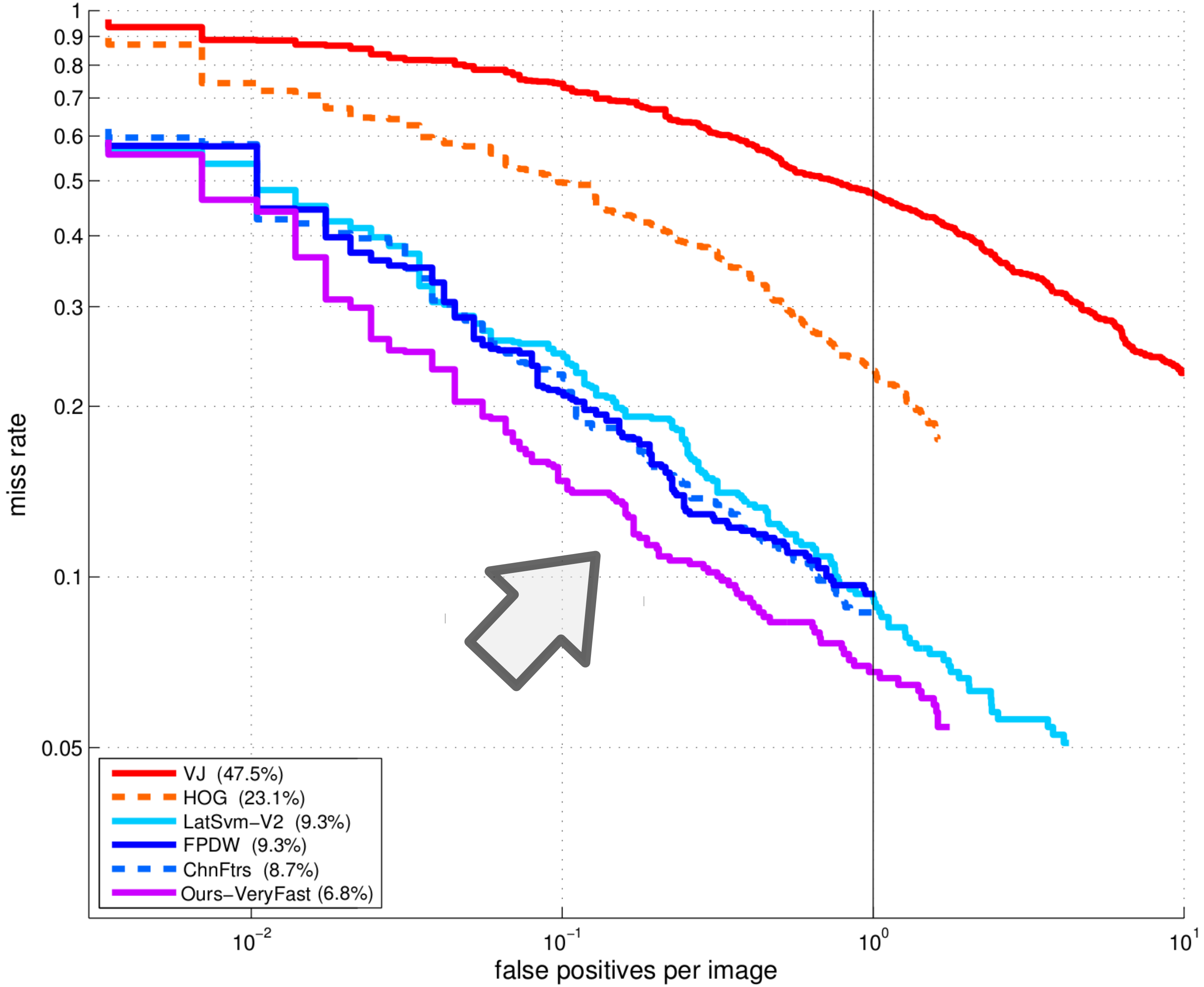


5 models,
1 image scale

50 models,
1 image scale


6 Orientation bins
Gradient magnitude
LUV color channels
Feature responses
Scale 0.5
Scale 1
Scale 2
Scale 4
Scale 8

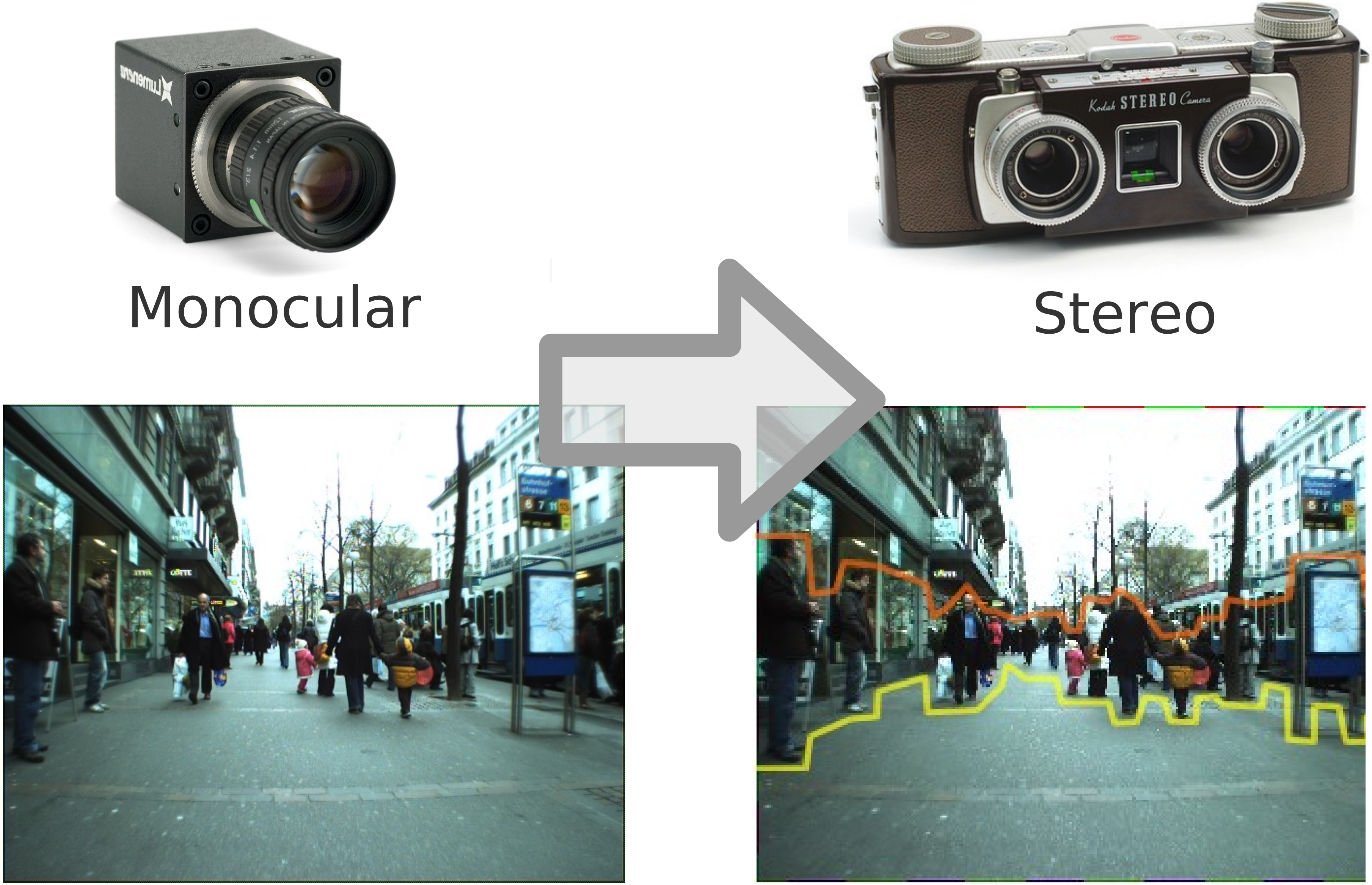
Detecting without resizing provides quality



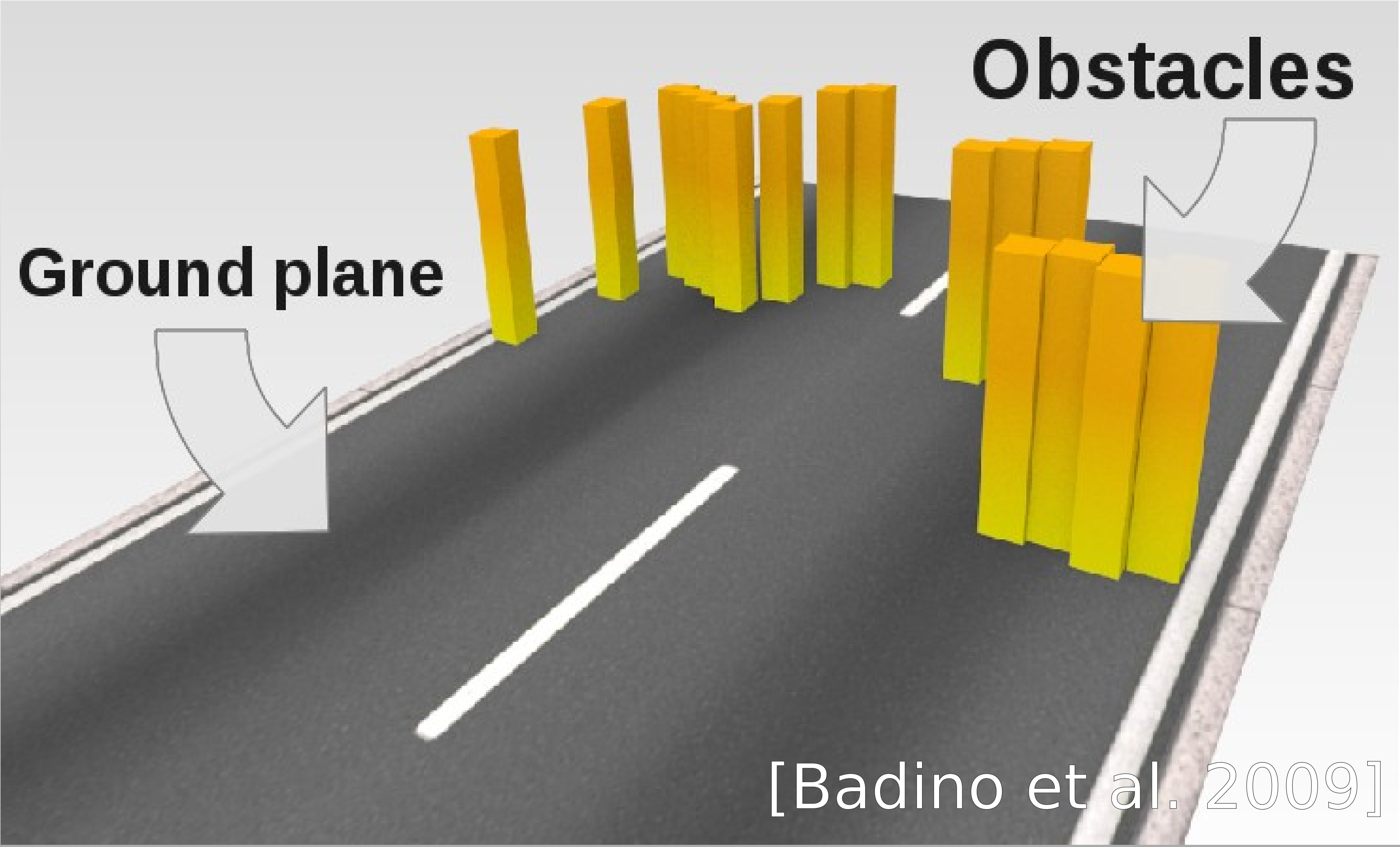
Detecting without resizing provides speed

- ~3x less time on features computation
- Avoids alternating between features and detection scores computation (relevant in practice)
- We reach **50 Hz** on GPU, 640x480 pixels x 55 scales

We want to use scene geometry to guide the detections

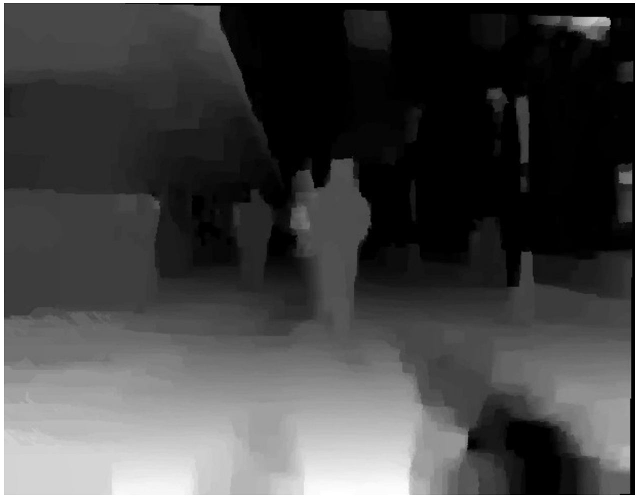


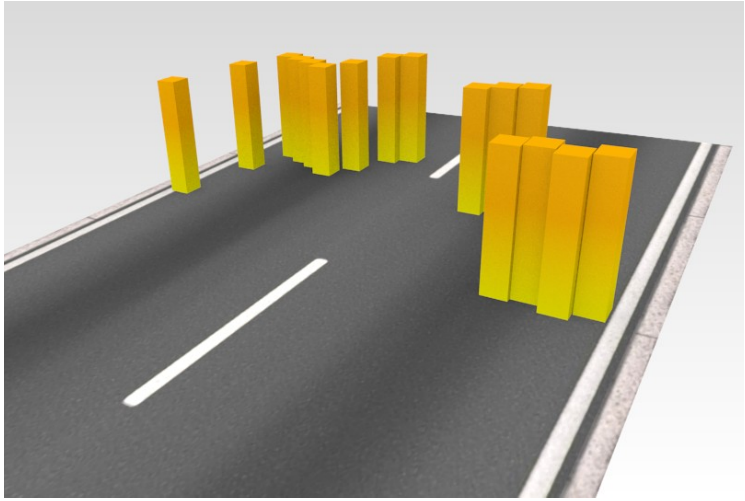
Stixel world



[Badino et al. 2009]

Depth maps are slow to compute

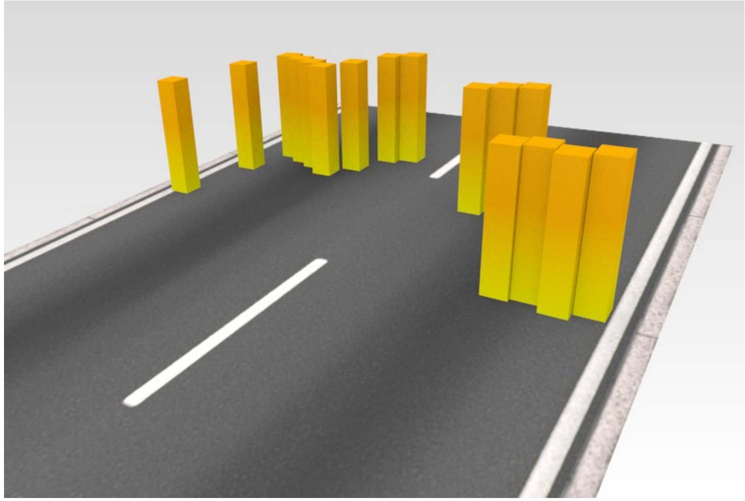




<50 Hz on CPU

Stixel world without depth map

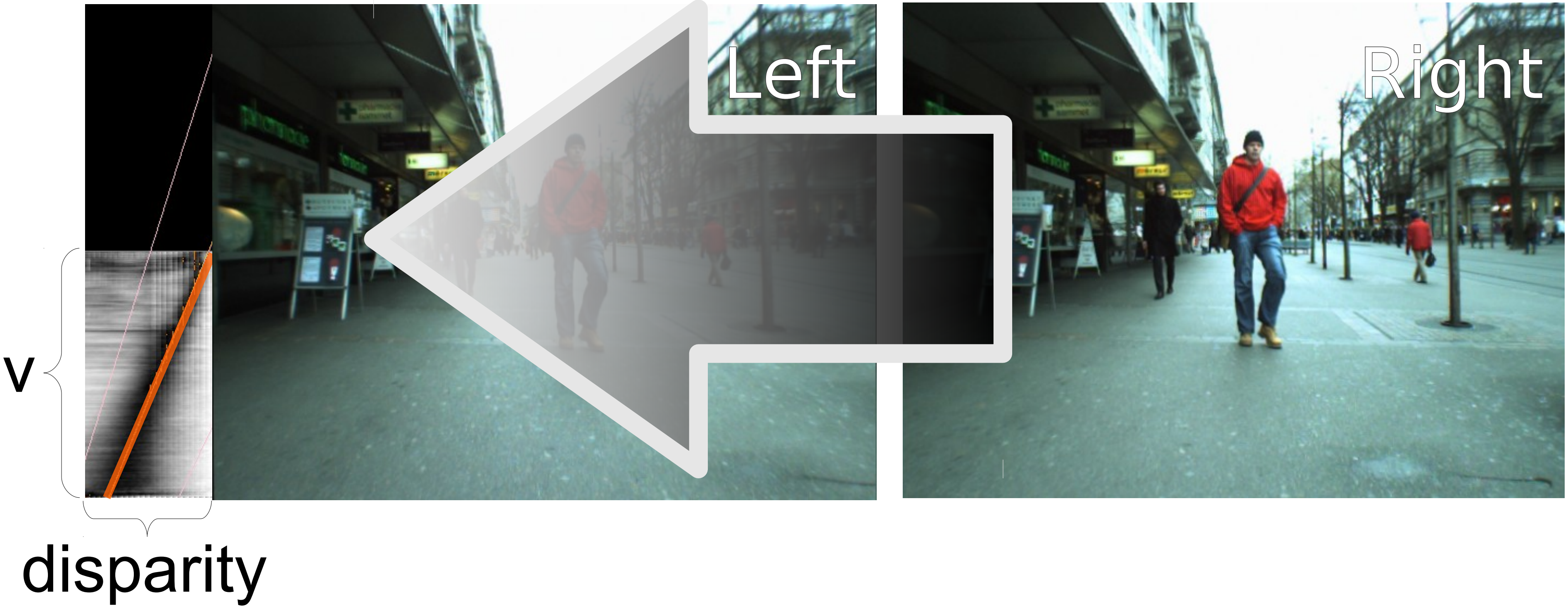
135 Hz on CPU



[Benenson et al. 2011]

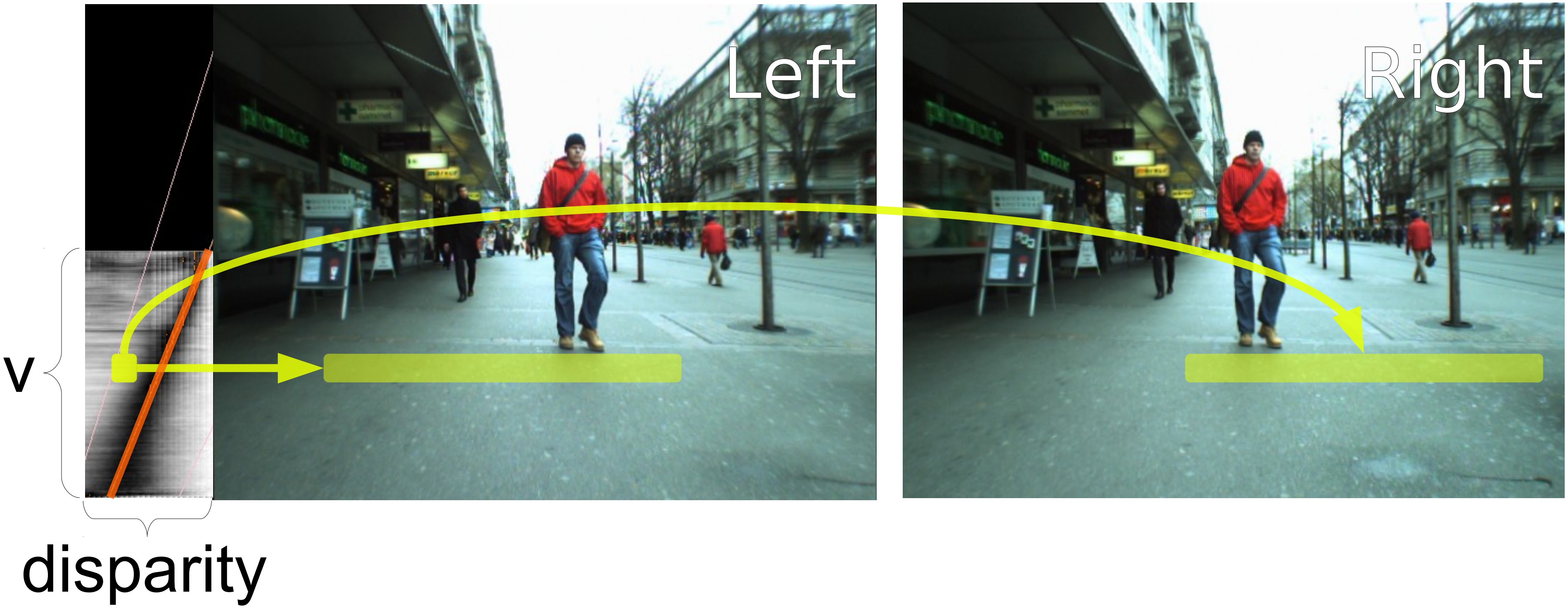
Stixel world without depth map

Stixel world without depth map



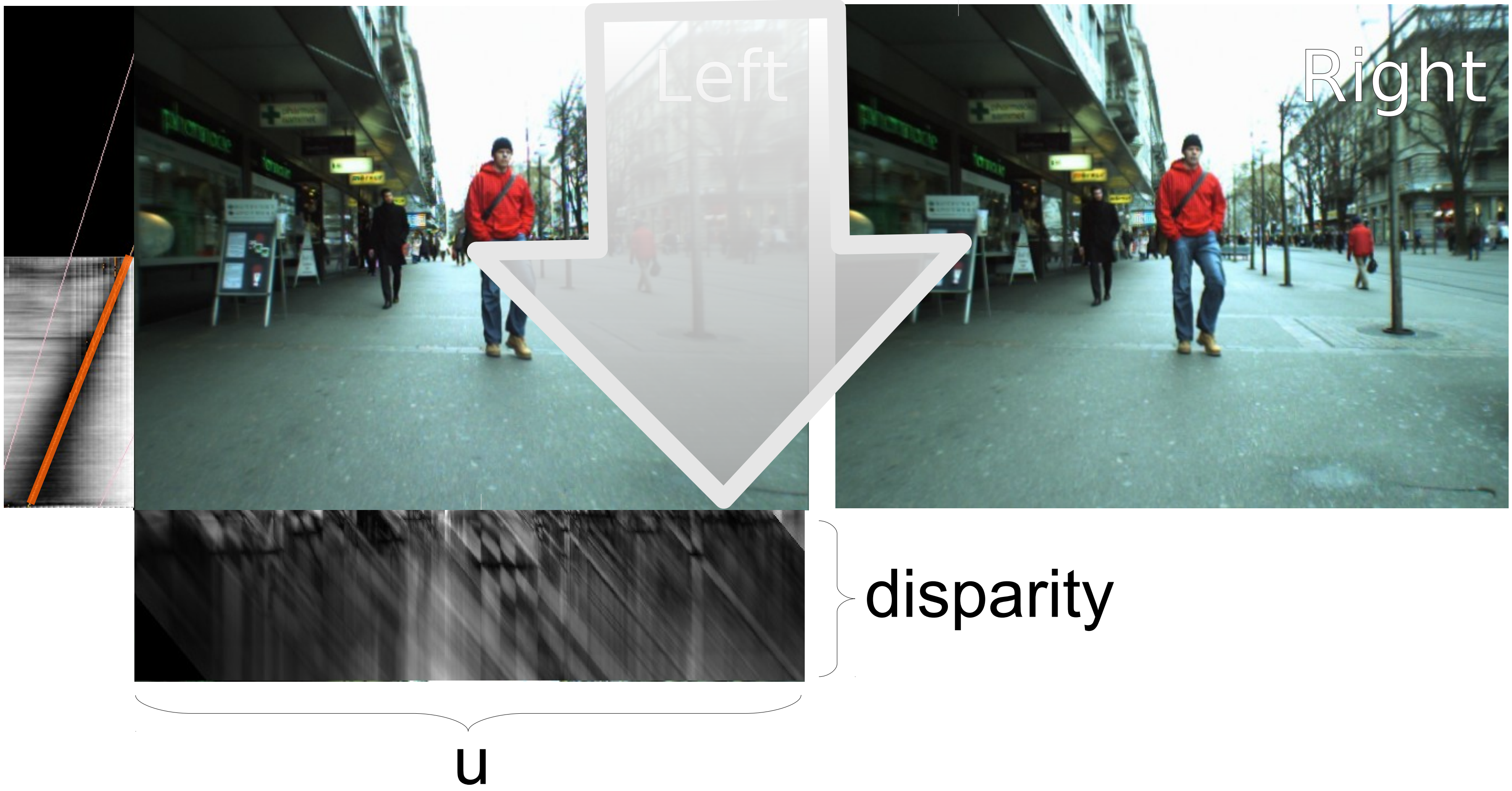
1) Ground plane estimation

Stixel world without depth map



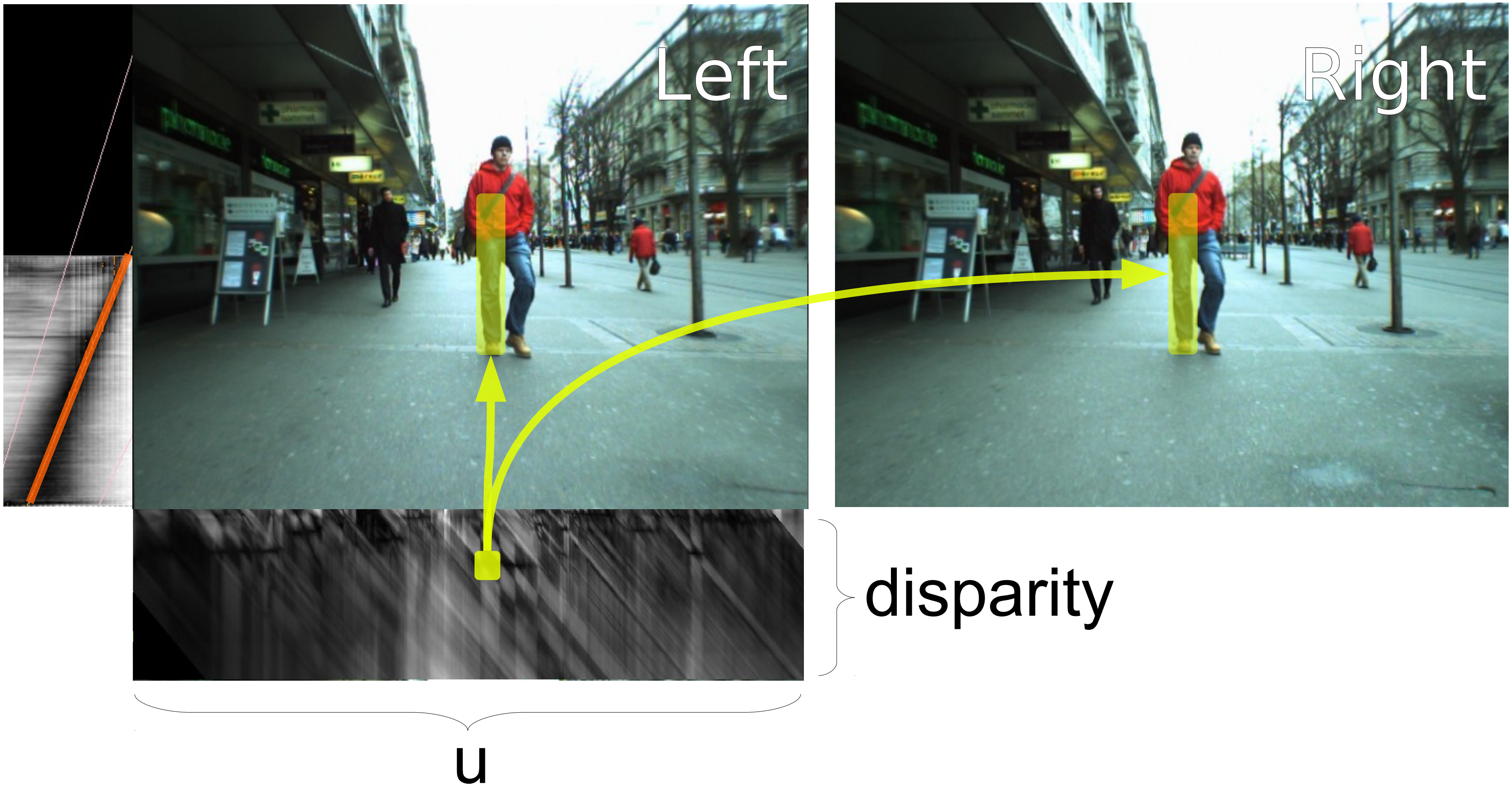
1) Ground plane estimation

Stixel world without depth map



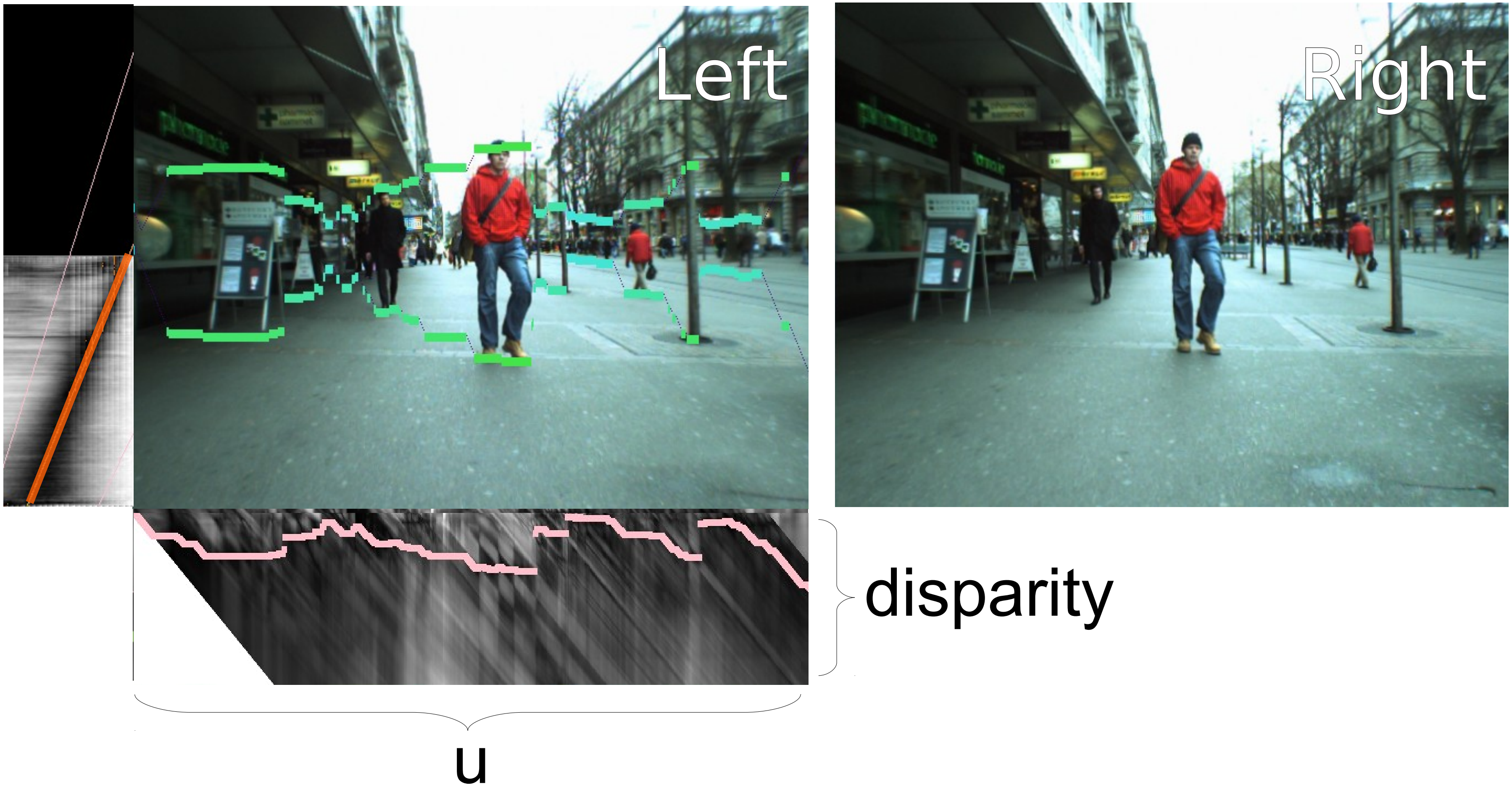
2) Stixel distance estimation

Stixel world without depth map



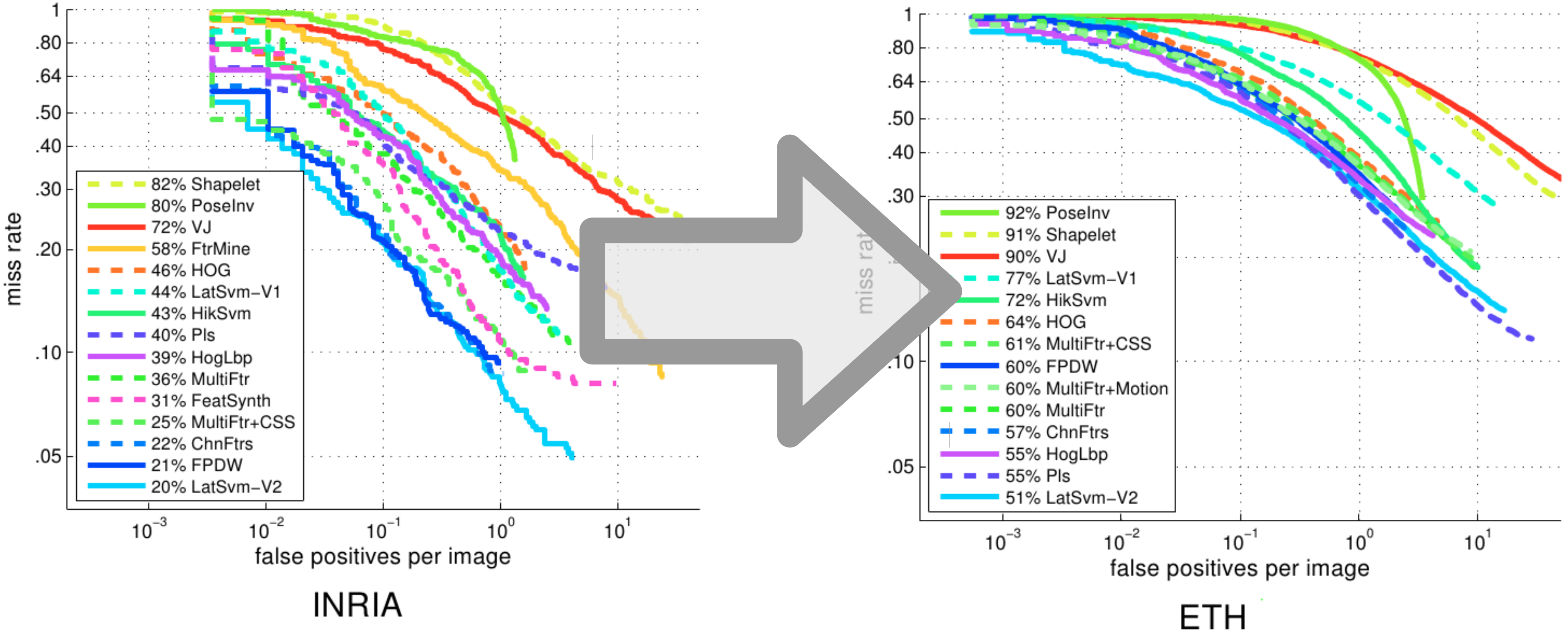
2) Stixel distance estimation

Stixel world without depth map



2) Stixel distance estimation @ 135 Hz CPU

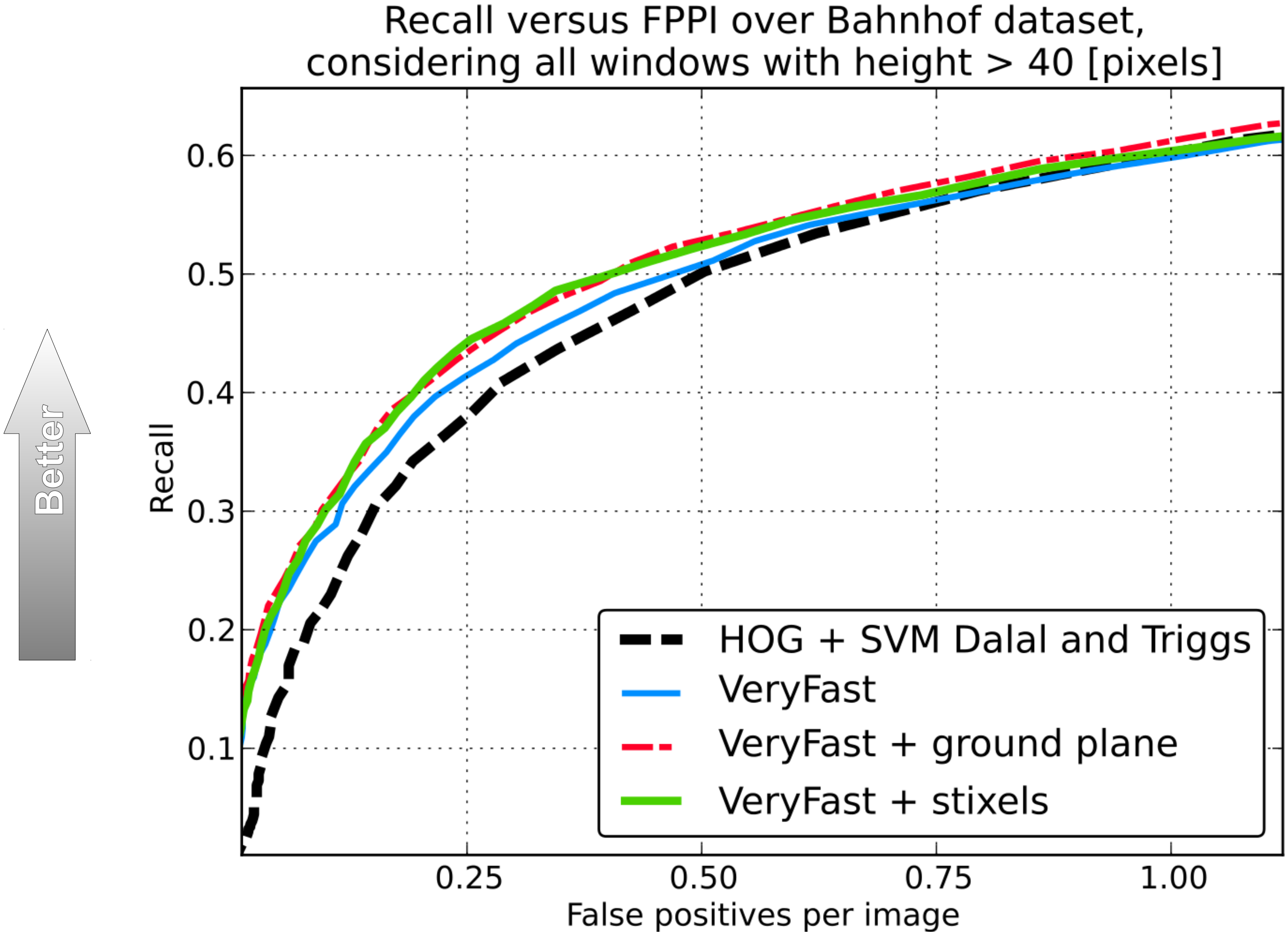
ETH's dataset results have less variance than INRIA's



Monocular

Stereo

Using stixels provides speed without quality loss



Detecting using stixels provides speed

- No geometry:
  - 640x480 pixels x 55 scales => 50 Hz on desktop
- Ground plane:
  - 640x60 pixels x 55 scales => 100 Hz on desktop (8x reduction in search space)
- Stixels:
  - 640x60 pixels x 10 scales => **135 Hz** on desktop (150 Hz GPU side, 135 Hz CPU side)
  - 44x reduction in search space
  - We reach *80 Hz on laptop*

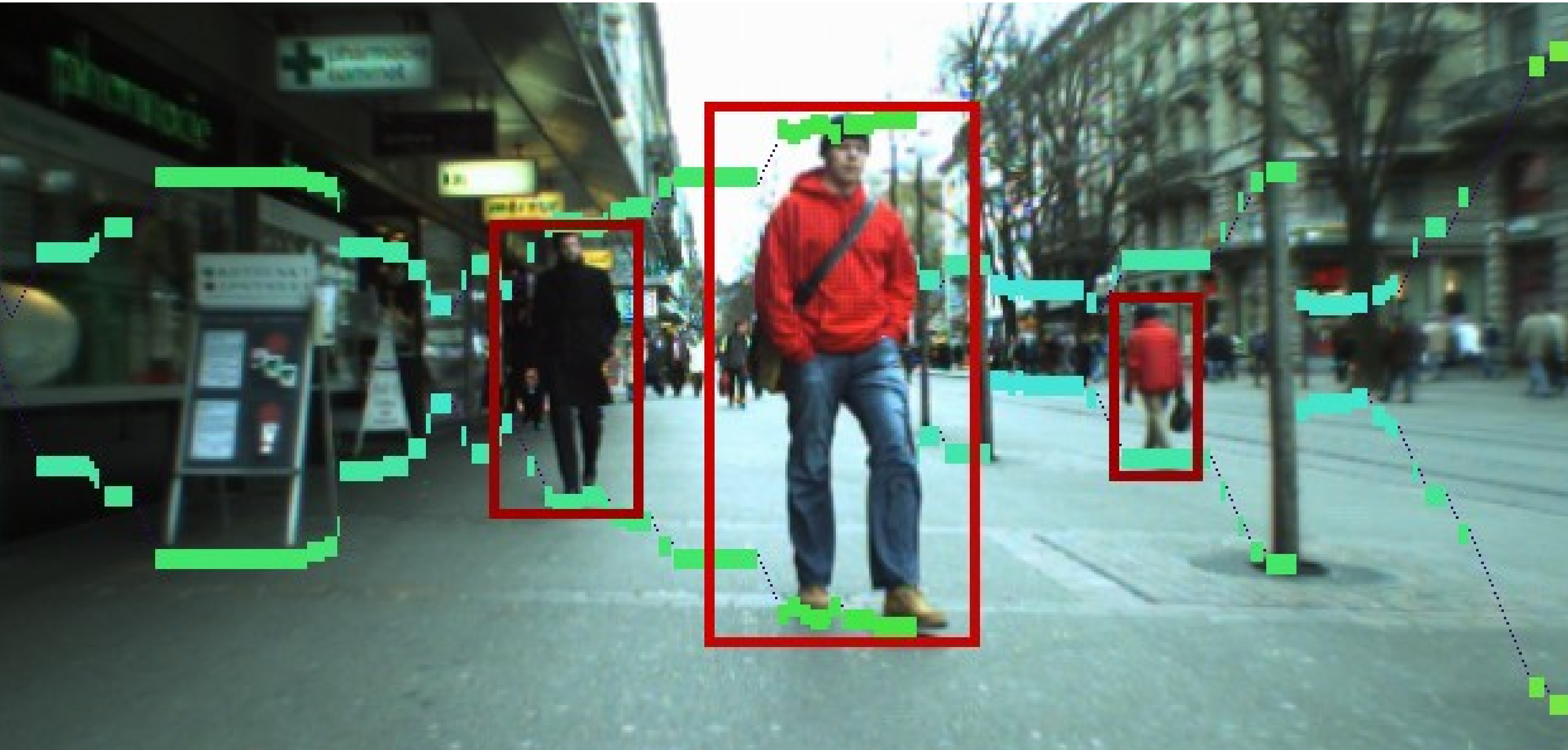
Win-win detector

- **High**est known **quality** for a single part detector
  (over the INRIA dataset, at camera ready time)

- 50 Hz in monocular mode,
  135 Hz in **stereo mode**, *80 Hz on a laptop*.

- 5x faster and 3x lower missrate
  than previous state-of-the-art, fastHOG.

No resizing + stixels
==
faster and better detections



Future work

- Transforming classifier seems useful:
  - Extension for different occlusions (submitted)
  - Extension for different point of views ?

Future work

- Transforming classifier seems useful:
  - Extension for different occlusions (submitted)
  - Extension for different point of views ?

- There is room for speed and quality improvements
  - (Original implementation was crude,
    work in progress version reaches ~170 Hz)

Future work

- Transforming classifier seems useful:
  - Extension for different occlusions (submitted)
  - Extension for different point of views ?

- There is room for speed and quality improvements
  - (Original implementation was crude,
    work in progress version reaches ~170 Hz)

- Building a part pased model
  on top of our VeryFast detector ?

?

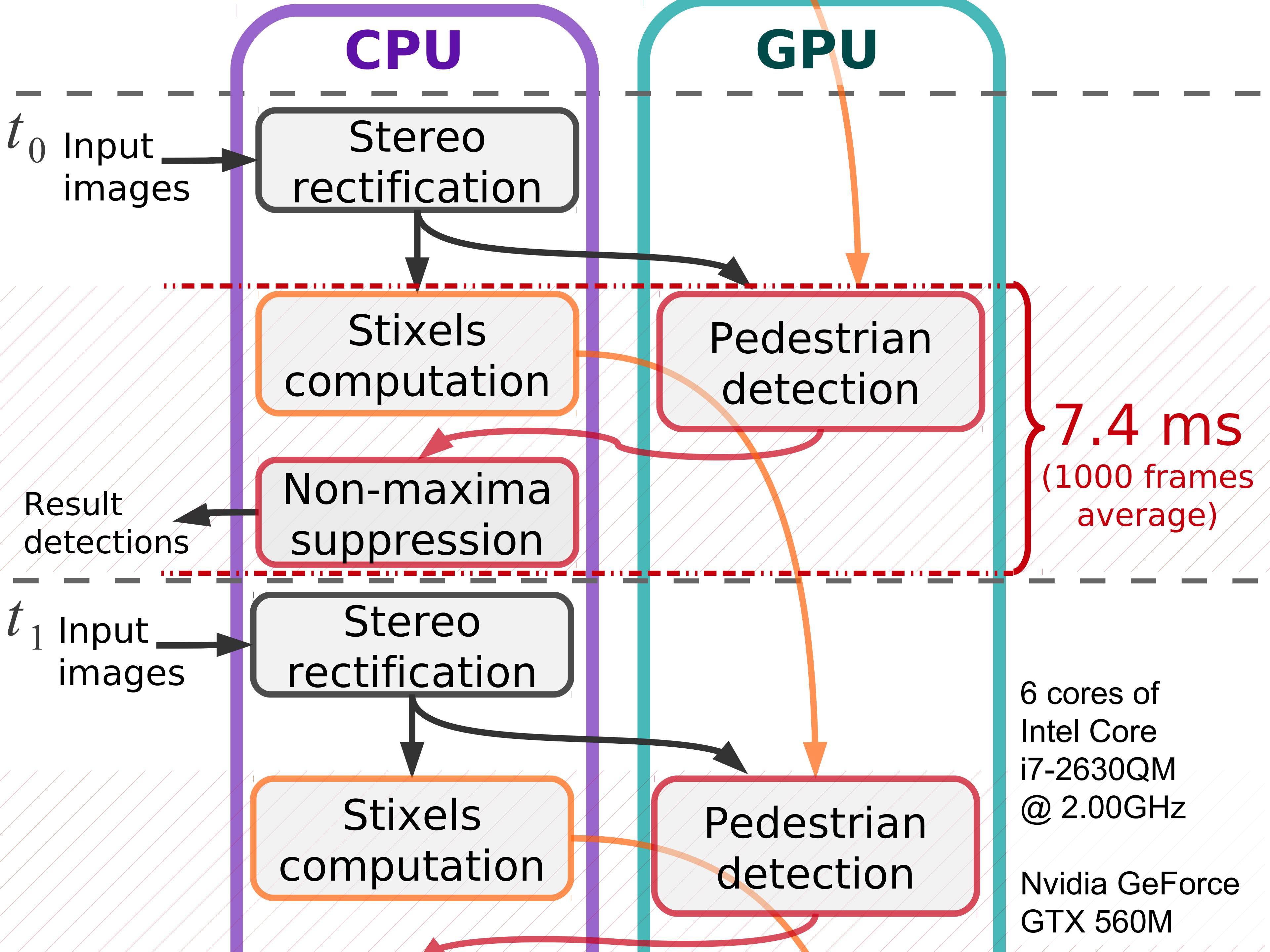
Rodrigo Benenson

http://rodrigob.github.com

Source code release on August 1st

KATHOLIEKE UNIVERSITEIT
LEUVEN

VISICS


CPU
GPU
t_0 Input images
Stereo rectification
Stixels computation
Pedestrian detection
Non-maxima suppression
Result detections
7.4 ms
(1000 frames average)
t_1 Input images
Stereo rectification
Stixels computation
Pedestrian detection
6 cores of Intel Core i7-2630QM @ 2.00GHz
Nvidia GeForce GTX 560M